Telefonaktiebolaget LM Ericsson
SE-126 25 Stockholm, Sweden

Inventor:

Knut Snorre Bach Corneliussen
Bygdøy alle 117 A
0273 Oslo
Norway

Multi-user applications in multimedia networks

## FIELD OF THE INVENTION.

The invention relates to the field of multi-user applications in systems of networked computers, and more particular to a multi-user computer system, method and arrangement employing multimedia call control, for alleviating problems of operation and administration of multi-user or real-time application programs in systems of networked computers.

## THE PROBLEM AREAS.

In systems with networked computers, it is often desirable to allow more than one user to interact with a single application at the same time (concurrently). Such applications are often called multi-user applications. Each multi-user application can be said to belong to one of the following two groups:

1) Multi-user applications with real-time requirements; and,
2) Multi-user applications without real-time requirements.

Typical examples of applications that belong to the first group are multimedia conferencing applications and multi-player games, while multi-user white boarding and word processors with document sharing are typical examples of applications belonging to the second group.

When enabling more than one user to interact with the same application, typically, the users are each provided with parts of the application, hereinafter referred to collectively as clients. The clients then communicate with the remaining parts of the application, hereinafter referred to collectively as the server. The physical location of the server can be a computer shared with one of the participating clients, which typically is the case for word processor sharing and games, or it can be a separate computer such as a dedicated server computer. Use of a separate computer is quite common when the shared application needs more resources than what is available at the location of any of the clients.

A protocol is used for information exchange between the client and the server. Although several standard protocols exist, customised protocols that are optimised for each type of application are commonly employed. The reason for this is that each type of application has its own, specific needs. A typical shared real-time application will often make use of small data packets to increase transfer speed, while non-real time applications will often make use of larger data packets to decrease the use of communication channel bandwidth for the information exchange.

Network games can be, as mentioned earlier, typical examples of multi-user applications with real-time requirements. In network games, each client runs most of the application locally. This means that the clients send only information to the server about the positions in the game and the current status of their respective players (the type of information, sent and received, is of course dependent upon the type of game). The server then co-ordinates and combines the information received from all clients and sends co-ordinated and combined information back to the respective clients. If only a small number of users, say, less than ten, is supported, then the server is often located with one of the clients. If, on the other hand, a large number of concurrent users are allowed, then the need for computer resources would be greater, and the server in such cases are often assigned separate hardware.

When, in a networked system, each such multi-user application is using its own protocol, this represents a significant problem to the administrator of these protocols, as it is difficult, and sometimes even impossible, for the administrator to perform common administration of the supported multi-user applications. In this context, administration is defined as:

- Methods for access control of who is allowed to communicate with the server
- Trace logs of usage
- Fault handling
- Administration of addresses and users
- All the needed logic to perform user billing of usage of the server
- Other types of administration

Yet another problem encountered in such situations is to enable the different multi-user application protocols to pass through a firewall. This is especially difficult with multi-user applications with real-time requirements, because such applications often use the User Datagram Protocol (UDP) as a transport protocol. Due to the connectionless nature of UDP, it is difficult to allow UDP based traffic to pass through a firewall and at the same time obtain good protection by the firewall.

Another problem related to using one of the standard call control protocols for multi-user server communication is that the existing means for transporting information, hence not session initiation information, in current solutions are based on codecs that are optimised for voice, video or other non real-time data transfer. For transport of real-time data, these codecs are not suitable.

Furthermore, it would be beneficial if all multi-user applications that operate in one domain could use the same communication protocol. If they all make use of the same communication protocol, administration problems (e.g. access control, trace logs, etc.) and communication problems (e.g. enable communication through a firewall) could be solved for the common protocol, and hence be used by all multi-user application servers.

## KNOWN SOLUTIONS AND PROBLEMS WITH THESE.

One suggested solution to the problem of administration is to implement separate support for administration of each type of application. The major problem with this method is, firstly, that for each new supported multi-user application the administration has to implement a new set of administration mechanisms, and secondly, that the administrator has to integrate the new set of administration mechanisms with existing administration for other multi-user applications.

Another suggested solution to the same problem is to support only multi-user applications that use a standardised protocol such as for example the Hyper-Text Transfer Protocol (HTTP). This, however, leads to other problems, as use of a single protocol will make it very difficult to make multi-user applications work in the network because of their different nature and their different resource requirements.

## OBJECTS OF THE INVENTION.

It is, therefore, an object of the invention to provide a solution to the problems outlined above, and which overcome the problems of the known solutions

## BRIEF DISCLOSURE OF THE INVENTION.

The present invention provides a system recited in the accompanying independent claim 1, a method recited in the accompanying independent claims 2, 8, 9 and 10, and an arrangement recited in the accompanying independent claim 7. Other advantageous features of the invention are recited in the accompanying dependent claims 3 – 6 and 11 - 14.

The present invention proposes a solution to solve the problem of administration of different multi-user applications by means of the H.323 standard according to ITU-T Recommendation H.323, 02/98 “Packet-based multimedia communications system", which is the standard mostly used for systems providing multi-media traffic today. Establishing and administrating connections between clients and their respective servers by means of H.323, according to the invention, provides the advantage of allowing a system that includes application specific protocols as well as one common standard protocol, namely the H.323.

BRIEF DESCRIPTION OF THE DRAWINGS.

Figure 1 is a block diagram representation of a simplified H.323 network example, illustrating client registration and authorisation.

Figure 2 is a block diagram representation of a simplified H.323 network illustrating set-up of a H.323 client-to-server network call and extended functions.

Figure 3 is a block diagram representation of a simplified H.323 network illustrating client-server information exchange according to the invention through a firewall.

Figure 4 is a schematic representation of an embodiment of a user data packet structure of the invention.

Figure 5 is a schematic representation of an embodiment of a control data packet structure of the invention.

Figure 6 is a sequence diagram illustrating an example of information exchange between server and client in an exemplary embodiment of a solution according to the invention.

Figure 7 is a sequence diagram illustrating an example of information exchange between server and client in another exemplary embodiment of a solution according to the invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS.

In the following, the present invention will be described by way of example and with reference to the accompanying drawings.

Referring to fig. 1, when the client is started, it first uses a known registration process of H.323 version 2 to be registered and authorised in the network. It should also be noted that, in the situation illustrated in fig. 1, the server is already registered in the H.323 network. When using H.323, the client side of the application and the server side of the application must both support the H.323 stack. Further, to have full service, the server must be running all the time. Through the registration process, the user is authorised (authorisation is new in version 2 of H.323). This means that the operator can decide who is allowed to contact the server. Up to this point, conventions and interactions in the network are according to the known steps of the H.323 version 2.

Now, referring to fig. 2, when the client initiates a call with the gaming server as the destination, the gatekeeper will check in the users profile, which is received from a User Handling Database (UHD), to see if the users is allowed to use the gaming server, denoted by Gaming Server Info in fig. 1. In order to fetch this data and to perform the evaluation, new functionality is added to a normal H.323 Gatekeeper. If the user is found to be allowed to "call" the gaming server, then the gatekeeper informs the client that he is allowed to make the call set-up as is typically done according to H.323.

Then the client starts the data channel of H.323 towards the server. This is allowed according to H.323, although it is not usually done this way as the procedure usually employed is to start voice and video channels. According to invention, the H.323 protocol is extended to support a new codec. This is shown below:

For the purpose of simplifying the explanation of the solution of the invention by way of example, in the following, H.323 and H.323 names and terms will be employed extensively. A new codec that is specialised for real-time data transfer has to be developed. In H.323 the new codec has to be identified in the ASN.1 syntax as described below:

```
DataApplicationCapability                         ::=SEQUENCE
  {
            application                           CHOICE
            {
                 nonStandard                 NonStandardParameter,
                 t120                             DataProtocolCapability,
                 dsm-cc                           DataProtocolCapability,
                 userData                         DataProtocolCapability,
                 t84                              SEQUENCE
                 {
                        t84Protocol
            DataProtocolCapability,
                        t84Profile                                  T84Profile
                 },
                 t434                             DataProtocolCapability,
                 h224                             DataProtocolCapability,
                 nlpid                            SEQUENCE
                 {
                 nlpidProtocol               DataProtocolCapability,
                        nlpidData                 OCTET STRING
                 },
                 dsvdControl NULL,
                 h222DataPartitioning             DataProtocolCapability,
                 ...,
                 t30fax                           DataProtocolCapability,
                 t140                             DataProtocolCapability
                 A new codec type                 DataProtocolCapability

            },
            maxBitRate                            INTEGER (0..4294967295),
            -- units 100 bit/s
            ...
  }
```

This basis for the ASN.1 code shown above is described in ITU-T Recommendation H.245, 02/98 "Control protocol for multimedia communication". The code above, being an amended code applicable to solutions according to H.245, shows how the Data Application field of the H.245 protocol is extended to accommodate the invention. In system operating according to the invention, such adaptation of H.245 is included in all clients, servers, and gatekeepers that has registered gaming servers to them (if the gatekeeper is routing h.245). In this context, the particular name of the new codec is not relevant, just that it is a new one. Any requirements for more than one new codec in a particular system will depend on the requirements defined for the communication between the client and the different types of gaming servers. What is important, though, is that a game server and all of its connected clients must provide support for the same codec type.

The new codec is designed in a simple fashion, meaning that it requires little overhead. In the following, some of the characteristics of the new codec are given:

- The codec uses RTP (Real-time transport Protocol) over UDP (User Datagram Protocol) to obtain real-time transport
- The codec includes mainly to types of messages: a) a data message, and b) a control message.
- The data message can be sent from the client or from the server. The control message is only sent from the server.

Referring to figure 4, an example of a data packet of the new codec will now be explained:

- In the Type field is an identifier defining the type of message is (e.g. 1 = data message, 2 =control message); which in this case is a data message.
- In the Protocol field is an identifier defining how the data in the rest of the message shall be interpreted. Note that there has to be a common understanding of the data format among client and server.
- In the Data field is included the data that is sent from the client or from the server

Referring now to figure 5, an example of a control packet of the new codec will be explained:

- In the Type field is an identifier defining the type of message is (e.g. 1 = data message, 2 =control message); which in this case is a control message.
- In the Protocol field is an identifier defining how the Control information in the rest of the message shall be interpreted. Note that there has to be a common understanding of the control format among client and server.
- In the Data field is included the control information that is sent from the server towards the clients. Examples of control information are how often data messages shall be sent from the client towards the server, and how often the server will send data messages towards the client.

Note that time-stamps and sequence numbers is not part of the codec messages, because this information typically can be obtained from the RTP header.

Now, with reference to the accompanying figures 6 and 7, and by way of example, information exchange in a client-server configuration in an embodiment of the invention will be explained. The referenced figures 6 and 7 generally show examples of the communication sequence between a client and a server. In the sequence examples shown, there are some common steps. The client initiates the communication by sending a “setup” message according to the standard call control protocol which has been selected; that is, generally, H.323 or SIP. Then the server signals accept of the incoming “setup” by sending an accept message according to the selected call control protocol. The call control part of the client then sends a suggested media set and address, including the new codec. Further, as shown in the examples, the suggested media set is accepted by the server by a message that also includes the media destination address to which the client is to send the media. At this point in the sequence, two different possibilities are available. One possibility is that the address is sent from the Call control towards the application part in both the server and the client, in which case it is the application on the client that sends the media using the new codec directly towards the application on the server. This first possibility is illustrated by the further parts of the sequence shown in figure 6. The other possibility is that the Call control on both the server and client sends a “start” message, or a similar kind of information, indicating that communication is now established between the server and the client. In this latter case, media sent in the new codec is first transmitted from the application towards the Call control and then from the Call control on one side over to the other Call control, and then on to the application. This other possibility is illustrated by the further parts of the sequence shown in figure 6. At the termination of a session, the client send a “close” message. However, closing can also be initiated by the server. The entity receiving the “close” message informs the application that the session is over, and responds to the “close” message by sending back an “accept” message.

Now, with reference to figures 4, 5, 6 and 7, the message types and their use will be explained by way of example. In accordance with a sequence as described above, the sever can first send a control message including information specifying the rate at which data is to be be sent form the client to the server, and possibly also information about the data type. In turn, the client sends data to the server at the specified rate and of the specified type, according to the scheme specified in the control message. Such control messages can be sent at any time during a session, in order for the server to specify different data rates and data types according to the needs of the application associated with the session.

With reference to the sequences explained above, it should be noted that the different possibilities illustrated by figures 6 and 7 also can be mixed or combined, in such a way that either the server party or the client party follows one of the sequences, while the other party follows the other sequence.

In H.323 networks with gatekeepers, all signalling must go through the gatekeeper. When the gatekeeper allows set-up and operation of a call, it can, according to known H.323 architecture and implementations, inform the normal charging system of that usage has started. A charging system can be added to a system, such as the system depicted in figure 1, in a number of different ways. A simple and effective way of accomplishing charging, is that the gatekeeper writes information related to call-setup and stop to an ASCII-file. A program can process this file by manual or automatic means at a later stage in time. A more advanced solution is to send Call Detail Records (CDR) to an external system. CDRs can include information about call start time, call stop time, activity, resources used, etc. The external system can then be made to automatically interpret these records and produce a cost of use (charging) to the end-user directly.

Further, as illustrated in figures 6 and 7, when seen in conjunction with the messages describe with reference to figures 4 and 5, when establishing the data channel, the client informs the server of which protocol to use for the data communication. For a system according to invention, this is to be the new codec as described above. This means that the applications themselves can use whichever protocol they desire, as long as it maps into the new codec type. During the H.323 set-up phase, both the client and the server also inform each other of ports on which they want to receive data, and of which transport protocol is to be used, such as e.g. whether they use TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). This information can further be used by a H.323 Proxy to enable the chosen data protocol to be transferred through a firewall, as illustrated in fig. 3. If an H.323 proxy is used, it also will be updated with the enhanced H.323 protocol.

Referring to fig. 3, an example is shown, wherein two clients communicate with the server. They use both the H.323 protocol, which is sent via the gatekeeper, and the chosen data protocol directly. When the data channel is established, the client informs the server of which protocol to use for the data communication. This means that the applications can use any preferred protocol. In figure 3 a firewall is also shown together with a H.323 proxy. The reason for including the proxy functionality is two fold. Firstly, it is quite common to have a firewall at every enterprise and ISP to protect their respective areas. Secondly, NAT (Network Address Translation) is often used by enterprises for sharing one single IP-address, and for not giving away information about IP-address for nodes located inside the domain of the enterprise. H.323 does not include support for NAT and proxies in itself, but by using the H.323 proxy, the IPT solution alleviates the limitations of the H.323 v2 standard related to communicating with endpoints located behind firewalls.

In accordance with the above, the H.323 Proxy contains the following functions:

- The H.323 proxies the RAS signalling (registration and admission), and replaces the internal IP addresses of the enterprise in the RAS messages with public IP addresses (NAT-Network Address Translation).
- The H.323 proxies the Q.931 signalling (call set-up), and replaces the enterprise internal IP addresses in the Q.931 message with public IP addresses.
- The H.323 proxies the H.245 signalling (media channel set-up), and replaces the enterprise internal IP address in the H.245 messages with public IP addresses. If the endpoint uses H.245 in a separate channel the H.323 proxy transforms this to a tunnelled H.245 since the IPT system always uses tunnelled H.245.
- The H.323 proxy controls the media streams that are set-up as a result of the H.245 signalling, and proxies the media streams (media stream NAT).

From the example described above, and as illustrated by figure 3, it should be noted that the chosen data protocol can be sent through a firewall when using a H.323 proxy.

The procedure of authenticating H.323 end-user in H.323 systems is specified in H.323. However, it is only specified how the user name and password can be sent from an end-user to the gatekeeper. To obtain true authentication, a means for checking the username against the password must be added to the system. On way of allowing this check to be accomplished is, as illustrated in the accompanying figures 1, 2 and 3, to add a database to the gatekeeper. This database will at least include a record for each user containing a username and a password. The gatekeeper will then check if the end-user exist in the database. If the end-user exists in the database, then the gatekeeper obtains the password of the user and checks to see if it matches the password received by the end-user. The database and its associated logic is in this solution referred to as the User Handling. The data that determines if the user is to be allowed to make a call can be added to the user record stored for each user in the database described above.

During the H.323 set-up phase, the client and the server also both inform each other of on which ports they want to receive data and of whether they are using the Transmission Control Protocol (TCP) or UDP. This information can be used by a H.323 Proxy to enable the chosen data protocol to be transferred through a firewall.

When the set-up phase is over, the respective client and server use the chosen protocol that is optimised for their needs to transfer data to each other.

When the session is over, the client closes the connections and informs the gatekeeper. The gatekeeper then informs the charging system that the usage of the server has stopped. If a client should become inoperable or a network failure should occur, then the system can also detect this because H.323 requires regular updates of the status of the "call". A correct record of time of usage is therefore guaranteed.

Although only a simple H.323 network is shown in the figures 1, 2 and 3 to simplify the drawings for the purpose of explaining the invention, the solution provided by the present invention will also work in large-scale H.323 networks or in networks having an architecture and/or operating according to similar call control protocols, such as for example the SIP protocol.

ADVANTAGES

By using H.323, the applications can easily be integrated with voice and video if they do not already have this support. This will give some application a new dimension without the need for making large changes the application required otherwise.

When using H.323, the client does not need to know the IP(Internet Protocol)-address of the server, as the H.323 supports more advanced address schemes like E-164 numbers, e-mail addresses or aliases.